eip ESCUELA INTERNACIONAL POSGRADOS

Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

Programación Avanzada Python

LECCIÓN 03

Programación y manejo de funciones

ÍNDICE

INTRODUCCIÓN

En esta tercera lección vamos a estudiar como trabajar con funciones. Para ello empezaremos viendo como definir una función.

Seguidamente, veremos como enviar valores y como retornar valores desde una función. Una parte fundamental para el envió de valores son los conceptos de argumentos y parámetros como veremos en esta lección.

Por último, estudiaremos como controlar errores dentro de una función.

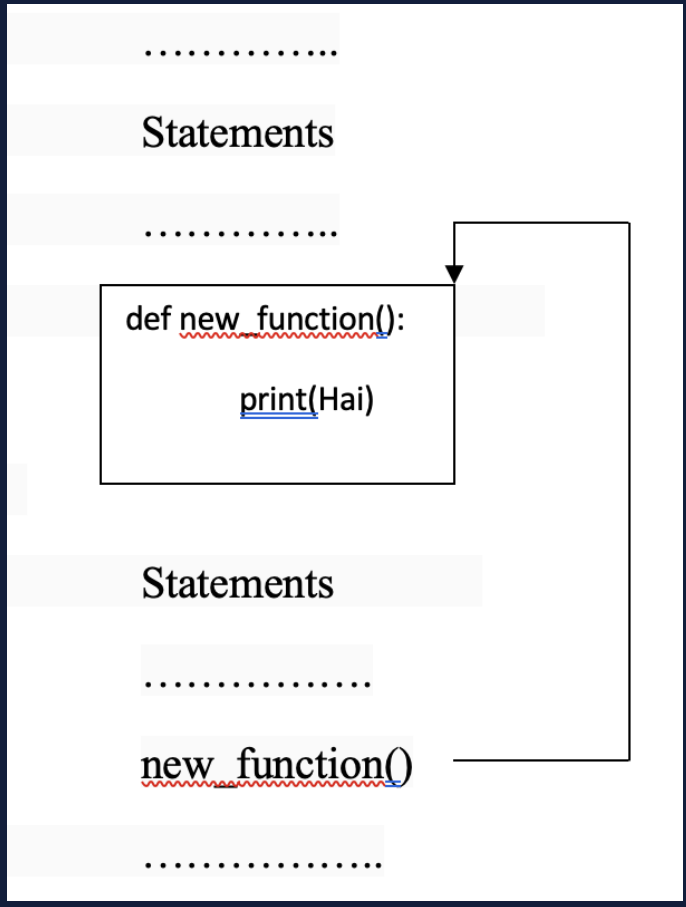
OBJETIVOS

Al finalizar esta lección serás capaz de:

1. Crear funciones en Python
2. Enviar y retornar valores en funciones
3. Trabajar con argumentos y parámetros
4. Controlar errores con excepciones

Funciones en Python

La función es un conjunto de códigos que se ejecutan cuando se le llama y realiza una determina acción. Las funciones se utilizan principalmente para hacer el programa más compacto y cuando un conjunto particular de códigos quiere ser reutilizado.



El nombre de la función se escribe con la palabra clave def, seguido del nombre de la función, el paréntesis y los dos puntos.

La llamada a la función sería:
Función_nombre()

```
def new_function():
  print("Hai")

new_function()
```

```
Hai
```

Envío de valores

Una función puede tomar los datos como entrada a través de sus paréntesis, estos valores se llaman parámetros. Los parámetros se declaran en el momento de la definición de la función.

```
def example_fun(name):
  print("hello",name)
```

```
example_fun('abc')
example_fun('qaz')
example_fun('pqr')
```

```
hello abc
hello qaz
hello pqr
```

Envío de valores

- No hay límite en el número de parámetros.
- Los valores del parámetro se envían de forma ordenada.

```
def fun_parameters(name, age):
  print('I am', name,"& my age is",age,)

fun_parameters('xyz',55)
```

```
I am xyz & my age is 55
```

Envío de valores

- Es posible dar los parámetros en diferente orden especificando el valor de cada parámetro.

```
def my_function(name3, name2, name1):
  print("The third one is " + name3)

my_function(name1 = "xyz", name2 = "abc", name3 = "def")
```

```
The third one is def
```

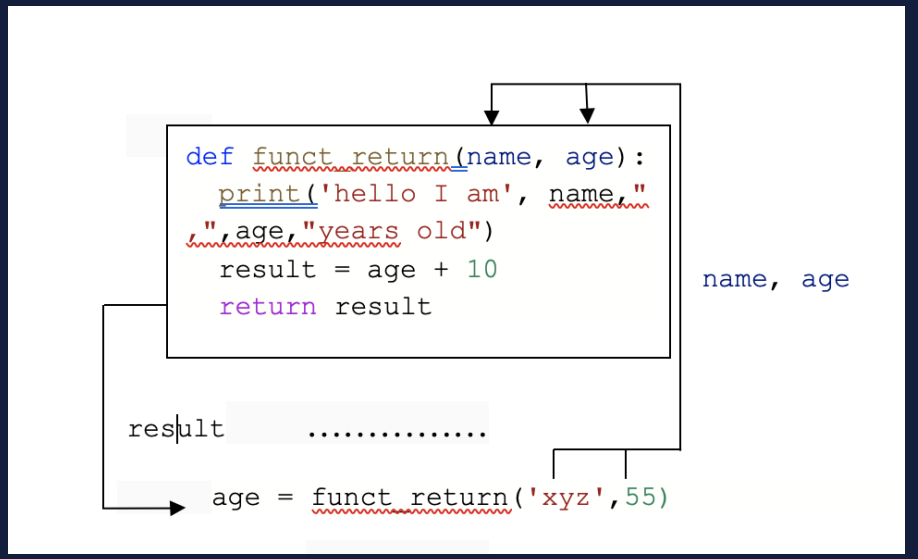
Retornando valores

Las funciones pueden devolver o retornar una salida. La palabra clave 'return' se utiliza en la función para devolver dichos valores

```
def funct_return(name, age):
  print('hello I am', name,",",age,"years old")
  result = age + 10
  return result
```

Retornando valores

Funcionamiento:




```
age = funct_return('xyz',55)
print("after 10 years, my age become", age)

hello I am xyz , 55 years old
after 10 years, my age become 65
```

Retornando valores con condiciones

Una función puede devolver el valor basándose en algunas condiciones:

```
def conditions(x,y):
  if x>y:
    return x-y
  else:
    return y-x
```

```
print(conditions(10,5))
```

```
5
```

Argumentos y parámetros

Parámetros - son los valores definidos en la función en el momento de la definición

Argumentos - son los valores dados por el usuario y pasados a las funciones

```
def fun_parameters(name, age):
  print('I am', name,"& my age is",age,)

fun_parameters('xyz',55)
```

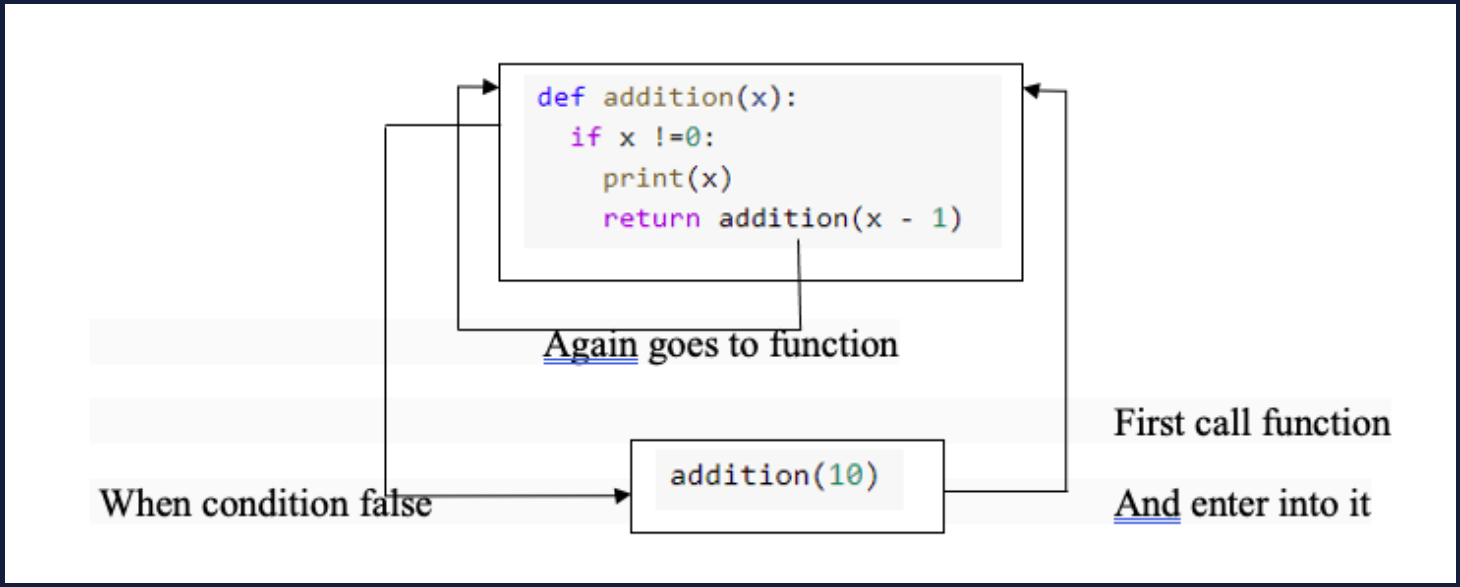
```
I am xyz & my age is 55
```

Parámetros - name y age se definen en la definición de la función

Argumentos - xyz & 55 son valores que se pasan como parámetro a la función.

Funciones recursivas

Si una función devuelve la misma función, se llama recursión. El control se mantiene como una rotación hasta satisfacer una condición o con una declaración de ruptura.



Excepciones

En Python, los posibles errores de una función pueden ser capturados usando algunos objetos que se llaman excepciones.

Sintaxis:

```
try:
        expresiones/declaraciones
except:
        print error
```

Excepciones

```
x = '5'
y = 5
try:
  x+y
except:
  print("Error")
```

```
Error
```

Finally

```
x = '5'
y = '5'
try:
  x+y
except:
  print("Error")
finally:
  print('No error')
```

```
No error
```

Assert

En Python, los errores en la lógica del programa pueden ser señalados con declaraciones de aserción. El error de aserción puede ser capturado con la palabra clave assert:

```
x = -2
assert (x > 0), "x is negative number"
```

CONCLUSIONES

1. Una función puede tomar los datos como entrada a través de sus paréntesis, estos valores se llaman **parámetros**.

2. Las funciones pueden devolver o retornar una salida, mediante la palabra reserva **return**.

3. Los posibles errores pueden ser capturados usando algunos objetos que se llaman **excepciones**.

MUCHAS GRACIAS POR SU ATENCIÓN

eip ESCUELA INTERNACIONAL POSGRADOS

rsanchezi@grupomainjobs.com

Rubén Sánchez Iruela
linkedin.com/in/ruben-sanchez-iruela-8156799a

twitter.com/eiposgrados

facebook.com/eiposgrados

instagram.com/eiposgrados